Lehrstuhl für Informatik 3
Rheinisch-Westfälische Technische Hochschule
Aachen
Univ.-Prof. Dr.-Ing. M. Nagl

Sommersemester 2007: Termine zur Vorlesung
„Grundgebiete der Informatik 2:
Algorithmen und Programmiertechniken“

Veranstaltung		Wochentag	Zeit	Ort	Beginn	Person
Vorlesung		Freitag	10:00–11:30	AM	13.04.	Prof. Nagl
Übungsgruppe	1	Montag	15:30–16:15	Be 114	16.04.	Hubertus
	2		16:30–17:15			Rehbaum
	3	Dienstag	17:00–17:45	Be 114	17.04.	Thorsten
	4		18:00–18:45			Fuhrmann
	5	Mittwoch	12:15–13:00	RS 4	18.04.	Philipp
	6		13:15–14:00			Kosse
	7	Mittwoch	12:15–13:00	RS 5	18.04.	Annegret
	8		13:15–14:00			Klein-Heßling
	9	Donnerstag	12:15–13:00	SG 413	19.04.	Tilo
	10		13:15–14:00			van Ekeris
C++-Betreuung		Montag	11:30–13:15	CIP-Pool	16.04.	wechselnd
		Donnerstag	8:30–10:30		19.04.	
Fragestunde		Freitag	14:30–	AH V	20.04.	Assistenten

Die Lage der Übungsräume kann dem Anhang des Vorlesungsverzeichnisses entnommen werden. In der Exkursionswoche vom 28.05.2007 bis zum 01.06.2007 finden keine Übungen statt. Die Übungsaufgaben der Vorwoche sind deshalb umfangreicher. Der CIP-Pool des Rogowski-Instituts für Elektrotechnik befindet sich im 2. Stock des Seminargebäudes. Dort stehen zu den angegebenen Zeiten Ansprechpartner zur C++-Programmierung zur Verfügung.

Die Anmeldung zu den Übungsgruppen zu dieser Veranstaltung kann am 13.04.2007 ab 13:00 Uhr bis 16.04.2007 (8:00 Uhr) auf der Webseite (`http://www-i3.informatik.rwth-aachen.de/ggdi-anmeldung.html`) vorgenommen werden.

Die Klausuren finden am 17. 07. 2007 und am 04. 09. 2007 statt. Genaueres wird auf unserer Webseite bekanntgegeben.

Lehrstuhl für Informatik 3
Rheinisch-Westfälische Technische Hochschule
Aachen
Univ.-Prof. Dr.-Ing. M. Nagl

Aachen, 13. 04. 2007

Nagl, Ranger, Wörzberger

Übungen zur Vorlesung „Grundgebiete der Informatik 2: Algorithmen und Programmiertechniken“

— Blatt 1 —

Allgemeines zu den Übungen

Die Aufgabenblätter werden immer freitags am Ende der Vorlesung verteilt. Die bearbeiteten Übungen müssen jeweils am darauffolgenden Freitag *vor* der Vorlesung abgegeben bzw. bis zum gleichen Zeitpunkt per E-Mail an `ggdi2-xxx@i3.informatik.rwth-aachen.de` geschickt werden (für xxx jeweils den Vornamen des Übungsgruppenleiters einsetzen, also z.B. ggdi2-thorsten@i3.inf...). Bitte schreiben Sie Ihren Namen und den Übungstermin bzw. den Namen Ihres Übungsleiters auf Ihre Lösungen, damit wir sie richtig zuordnen können. Korrigierte Übungsblätter werden in den Übungsgruppen der nachfolgenden Woche zurückgegeben. Die Übungsgruppen finden ab nächster Woche (KW 16) statt.

Zur Bearbeitung der Übungen müssen Arbeitsgruppen von drei Teilnehmern gebildet werden. Die Übungstermine sollten für Fragen aller Art genutzt werden.

Für einige Aufgaben ist ein Programm in der Programmiersprache C++ zu erstellen. Als Lösung ist ein syntaktisch korrektes Programmlisting abzugeben *und* dem Übungsgruppenleiter elektronisch zu übermitteln (per E-Mail). Dieses Programm ist mit einem Compiler zu überprüfen. Dazu können eigene Compiler oder der des CIP-Pools verwendet werden. Dort kann während der normalen Öffnungszeiten programmiert werden. Für diese Veranstaltung reservierte und betreute Zeiten im CIP-Pool sind Montags zwischen 11:30 und 13:15 sowie Donnerstags zwischen 8:30 und 10:30 Uhr. Bitte stellen Sie Fragen zu C++ oder zu den Aufgaben **nicht** dem Beratungspersonal des CIP-Pools, sondern nur den Betreuern der Übungsgruppen.

Verweise auf frei verfügbare C++-Compiler finden sich (neben anderen Informationen zur Vorlesung) auf `http://www-i3.informatik.rwth-aachen.de/ggdi2`.

Um Ihren Lernerfolg zu überprüfen, bieten wir Ihnen an, zu dieser Veranstaltung einen Übungsschein zu erwerben. Jeder Aufgabe ist eine Punktzahl zugeordnet. Um sich für den Schein zu qualifizieren, müssen Sie über das Semester hinweg mindestens 50% der Gesamtpunktzahl aller Aufgaben erreichen. Zusätzlich müssen Sie mindestens eine Aufgabe in Ihrer Übungsgruppe vorrechnen. Ihr Übungsgruppenleiter wählt für jede Aufgabe unangekündigt einen Teilnehmer zum Vorrechnen aus. Nur wer dann anwesend ist und vorrechnet, hat ein Anrecht auf den Schein. Für Studierende, die keinen Schein erwerben möchten, ist die Teilnahme an der Übung und das Vorrechnen freiwillig.

Unter `news://news.rwth-aachen.de/rwth.elektrotechnik.ai2` ist eine Newsgroup zur Vorlesung eingerichtet. Hier können Sie untereinander Erfahrungen austauschen.

1. Aufgabe *Aufbau von EBNF-Regeln*: *(13 Punkte)*

In der Vorlesung haben Sie zur Syntaxbeschreibung (Grammatik) von Programmiersprachen EBNF-Regeln kennengelernt. EBNF-Regeln bestehen aus Terminalsymbolen und Nichtterminalsymbolen, die mit bestimmten Operatoren verknüpft werden. Terminalsymbole sind Elemente der beschriebenen Sprache, beispielsweise Schlüsselwörter wie class oder if in C++. Nichtterminalsymbole definieren einerseits komplexe Spracheinheiten, andererseits helfen sie die Syntaxbeschreibung zu strukturieren.

(a) Als Konstruktionsmöglichkeiten in EBNF-Regeln gibt es Sequenz, Option, Wiederholung, Alternative und Rekursion. Geben Sie für jede dieser Konstruktionsarten drei Regeln (beziehungsweise zwei rekursiv definierte Nichtterminale für die Rekursion) aus der C++-Syntax im Anhang I des Skriptes an, in denen sie vorkommen. *(3 Punkte)*

(b) Gegeben ist folgende vereinfachte Grammatik für if-Statements in C++.

ifStatement	::=	if (*expression*) { *statementList* } **{** else if (*expression*) { *statementList* } **}** **[** else { *statementList* } **]**	(1)
expression	::=	*simpleExpression* **[** *relation simpleExpression* **]**	(2)
relation	::=	== | != | < | <= | > | >=	(3)
simpleExpression	::=	**[** + | – **]** *factor* **{** *addOperator factor* **}**	(4)
factor	::=	*number* | *ident* | (*expression*) | ! *factor*	(5)
statementList	::=	*statement* **{** *statement* **}**	(6)
addOperator	::=	+ | – | || | &&	(7)
statement	::=	*ifStatement* | cout << *string* ;	(8)

Hier soll *number* eine beliebige Ziffernfolge, *string* eine in Anführungszeichen eingeschlossene beliebige Zeichenfolge und *ident* eine Buchstabenfolge (Identifier) sein.

Leiten Sie die folgenden Statements aus *ifStatement* ab. *(4 Punkte)*

- `if (spielEnde) { cout << "Fertig"; }`
- ```
  if (s+k == 21) {
     cout << "Gewonnen!";
  } else if (s+k < 21) {
     cout << "Weiter";
  } else {
     cout << "Verloren";
  }
  ```

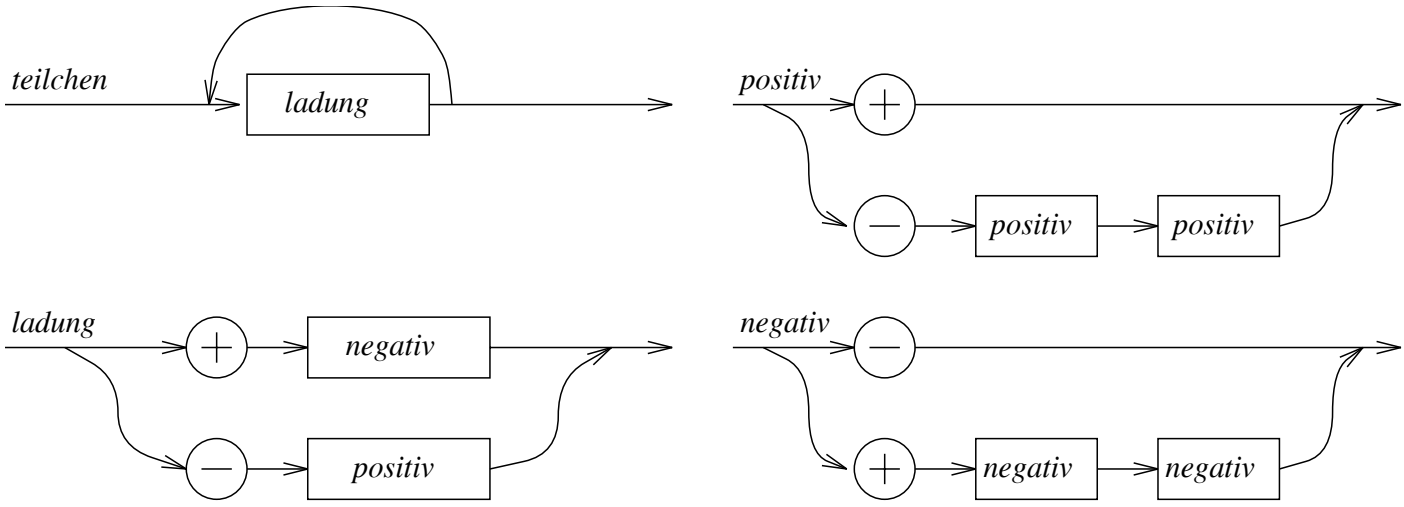
(c) Verändern und erweitern Sie die Grammatik aus (b) so, daß in Ausdrücken auch Multiplikation und Division verwendet werden kann. Sie können dazu ein neues Nichtterminal *term* einführen und in Regel (4) statt *factor* verwenden. Stellen Sie sicher Sie, daß ihre erweiterte Grammatik „Punkt- vor Strichrechnung“ beachtet.

(6 Punkte)
  ```

2. Aufgabe *Syntaxdiagramme*: *(11 Punkte)*

Statt EBNF-Regeln kann man für eine Syntaxbeschreibung auch Syntaxdiagramme verwenden. Syntaxdiagramme sind insbesondere für Ungeübte leichter zu verstehen, sind dafür andererseits weniger kompakt als EBNF-Regeln.

Gegeben seien die folgenden Syntaxdiagramme:



(a) Man kann EBNF-Regeln systematisch in Syntaxdiagramme übersetzen. Geben Sie dazu für Sequenz, Wiederholung, Option und Alternative Syntaxdiagramme an, die als Basisbausteine bei der Übersetzung dienen können. Warum haben wir die Rekursion ausgelassen? *(3 Punkte)*

(b) Übersetzen Sie die Diagramme in EBNF. *(4 Punkte)*

(c) Finden Sie eine Grammatik (in EBNF oder Syntaxdiagrammen), mit der die gleichen Worte erzeugt werden, die aber weniger Regeln hat. *(4 Punkte)*

3. Aufgabe *Mein erstes Programm*: *(15 Punkte)*

Machen Sie sich mit Editor und Compiler vertraut! Frischen Sie ihre Kenntnisse aus den Versuchen 4–8 aus dem Praktikum "Informatik I" anhand der folgenden Aufgaben auf:

(a) Schreiben Sie ein Programm, das die Namen der Mitglieder Ihrer Gruppe ausgibt. Übersetzen und starten Sie es. (Dieser Quelltext braucht nicht abgegeben zu werden.)

(b) Schreiben Sie ein Programm, das Zahlen von der Standardeingabe liest, bis diese endet und die Gesamtsumme ausgibt. Das Programm braucht bezüglich der Eingabe nicht besonders robust zu sein. (Dieser Quelltext braucht nicht abgegeben zu werden.)

(c) Teilen Sie Ihr Programm aus (b) in mehrere Quelltexte auf: `main.cc` (Leseschleife), `summe.cc` (Aufsummieren und Abfragen) und `summe.h` (Deklaration der Funktionen in `summe.cc`). *(6 Punkte)*

(d) Lesen Sie in der Dokumentation und schreiben Sie ein Programm unter Verwendung der Funktionen `time`, `localtime` und `asctime`, das die Sekunden seit Anfang 1970 sowie die aktuelle Zeit in der voreingestellten Zeitzone anzeigt. *(9 Punkte)*

Abgabe: 20. 04. 2007 vor der Vorlesung.